INTERFERENCE-AWARE DELTANET FOR STREAMING DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers excel at sequence modeling through attention, but their quadratic compute and growing key-value cache make them costly for long-range streaming tasks. Linear attention variants, such as DeltaNet, address this by reformulating attention as a recurrent memory equipped with an error-corrective update rule. However, we show that DeltaNet's update rule induces unintended interference on past-key responses, as the correction signal leaks into every past key that is not orthogonal to the current one. We propose Interference-Aware DeltaNet (IA-DeltaNet), which generalizes the delta rule by decoupling the key used for error evaluation from the direction along which the memory is updated. We formulate the choice of write direction as a constrained quadratic program that minimizes the total squared interference on past-key responses while exactly preserving DeltaNet's current-key correction, and derive its closed-form solution as a preconditioned key obtained from the inverse uncentered covariance of past keys. To ensure scalability, we introduce a diagonal approximation of the past-key covariance, preserving the per-step efficiency and chunk-parallelizability of the original architecture. On multi-query associative recall, interference-aware writing improves retention under high associative load. It also improves over recurrent-memory baselines as a bounded memory in a streaming VideoLLM, and substantially mitigates DeltaNet's degradation under prolonged recurrent-state propagation in language modeling. These results indicate that explicitly accounting for past-key geometry offers an effective design principle for associative memory in linear-attention architectures.

1 INTRODUCTION

Transformers with softmax attention dominate sequence modeling, yet their quadratic compute and growing key-value cache make them costly for long or continuously arriving inputs. Linear attention (Katharopoulos et al., 2020) addresses both costs by reformulating the attention as a recurrence over a fixed-size matrix-valued memory state, achieving linear-time compute and constant-memory inference. This recurrent formulation is particularly attractive for streaming data, where tokens arrive on the fly, the past cannot be revisited, and memory must remain bounded.

The recurrent state, however, must compress an arbitrarily long history into a finite memory, creating a tension between encoding new associations and preserving prior knowledge. In linear attention, additive accumulation simply aggregates key-value outer products; consequently, any retrieval is inherently corrupted by interference from overlapping keys. DeltaNet (Schlag et al., 2021) addresses this by employing an error-corrective delta-rule update. By selectively attenuating the existing memory response at the current key before writing the new value, it achieves significantly higher fidelity in associative recall tasks.

Despite this improvement, we formally demonstrate that DeltaNet continues to suffer from inherent past-key interference. Since at most d_k directions are mutually orthogonal in a d_k-dimensional space, writing the correction signal along $\boldsymbol{k}_t$ inevitably perturbs overlapping past keys, leading to cumulative interference as the stream progresses. This indicates that the fundamental limitation stems not from the error-correction mechanism itself, but from the dual role of $\boldsymbol{k}_t$ as both the read address for error evaluation and the write direction for memory updates.

To address this bottleneck, we propose Interference-Aware DeltaNet (IA-DeltaNet), a linear attention framework with an interference-aware error-corrective update that explicitly decouples the read and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

write roles of the key. While we retain $\boldsymbol{k}_t$ as the read address for evaluating prediction error, we introduce a distinct write direction $\boldsymbol{u}_t$ designed to minimize interference with past-key responses. We formulate this as a quadratic program subject to a linear constraint that preserves the exact current-key correction of DeltaNet. This optimization yields a closed-form solution in which an interference-aware direction is obtained by preconditioning $\boldsymbol{k}_t$ with the inverse uncentered covariance of historical keys. Notably, our approach generalizes DeltaNet under isotropic key geometry and is theoretically guaranteed to reduce total interference compared to DeltaNet.

However, computing this exact solution requires per-step operations on a full matrix, incurring an $O(d_k^2)$ overhead even with efficient incremental updates via the Sherman–Morrison formula. To mitigate this, we introduce a diagonal approximation that replaces the full covariance with a diagonal state, maintained as an element-wise prefix sum of squared keys. This approximation preserves the functional form of the optimal solution by suppressing heavily-used dimensions and renormalizing to satisfy the current-key correction constraint, while adding only $O(d_k)$ complexity per step. Furthermore, since the diagonal state can be computed as a simple prefix sum, the resulting write directions remain compatible with chunk-parallel training via associative scans.

We evaluate IA-DeltaNet on associative recall, streaming video understanding, and language modeling. On multi-query associative recall, interference-aware writing improves retention as associative load grows. The gains extend to needle-in-a-haystack retrieval in text and video. As a bounded recurrent memory in a VideoLLM, IA-DeltaNet improves streaming video understanding, and in language modeling it substantially mitigates DeltaNet's degradation under prolonged state propagation. Further analyses show reduced write interference and broader use of the write-direction subspace, and characterize when the diagonal approximation matches the exact solution. Together, these results suggest that explicitly accounting for past-key geometry is a simple yet effective principle for designing memory updates in linear-attention architectures.

Our contributions are summarized as follows:

- We provide a formal analysis of memory interference in DeltaNet, demonstrating how updating along the current key direction inevitably perturbs existing associations in the recurrent state.
- We introduce IA-DeltaNet, which decouples the read address from the write direction within the delta-rule update, and derive an optimal write direction in closed form that minimizes past-key interference while preserving current-key correction.
- We propose a diagonal approximation of our rule that maintains the efficiency of linear attention and enables chunk-parallel training via associative scans.
- We validate IA-DeltaNet on associative recall, streaming video understanding, and language modeling, and analyze how it reduces write interference.

2 Background

This section reviews linear attention (Katharopoulos et al., 2020) and DeltaNet (Schlag et al., 2021) to establish the necessary context for analyzing memory updates and the resulting interference.

2.1 Linear Attention as a Recurrent Memory

Let $\boldsymbol{q}_t \in \mathbb{R}^{d_q}$, $\boldsymbol{k}_t \in \mathbb{R}^{d_k}$, and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$ denote the query, key, and value vectors at time step t, respectively. Linear attention (Katharopoulos et al., 2020) replaces the softmax operation with a linear kernel similarity. For simplicity, we use a dot-product kernel, $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_i) = \boldsymbol{k}_i^\top \boldsymbol{q}_t$, and assume ℓ_2-normalized keys. By associativity, causal linear attention is expressed in the recurrent form:

$$\boldsymbol{o}_t = \sum_{i \le t} \boldsymbol{v}_i \left(\boldsymbol{k}_i^\top \boldsymbol{q}_t\right) = \left(\sum_{i \le t} \boldsymbol{v}_i \boldsymbol{k}_i^\top\right) \boldsymbol{q}_t = \boldsymbol{S}_t \boldsymbol{q}_t, \qquad \boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top. \tag{1}$$

Here, $\boldsymbol{S}_t \in \mathbb{R}^{d_v \times d_k}$ is a matrix-valued memory state storing an accumulation of past key-value associations. This formulation decouples linear attention into a memory update and a memory

readout. Instead of maintaining a key-value cache that grows with sequence length, the model propagates this fixed-size state, enabling constant-memory inference.

The compression inherent in this fixed-size state creates a fundamental tension between storing new information and preserving the existing memory response at the corresponding key. After the additive update writes the current association $\boldsymbol{v}_t\boldsymbol{k}_t^\top$, probing the memory with the current key $\boldsymbol{k}_t$ yields:

$$\boldsymbol{S}_t\boldsymbol{k}_t = (\boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top)\boldsymbol{k}_t = \boldsymbol{v}_t + \sum_{i<t} \boldsymbol{v}_i(\boldsymbol{k}_i^\top\boldsymbol{k}_t). \tag{2}$$

The first term is the target value, while the second term represents the pre-existing response at $\boldsymbol{k}_t$. This latter term is the interference induced by previously stored values, weighted by their similarity to the current key. Consequently, the memory response at $\boldsymbol{k}_t$ typically deviates from $\boldsymbol{v}_t$ unless the current key is orthogonal to all preceding keys.

This limitation originates from the bounded geometry of the key space. In d_k dimensions, at most d_k directions are mutually orthogonal; thus, new keys inevitably overlap with previous ones as the sequence progresses. Simple additive accumulation offers no mechanism to suppress this pre-existing response, leading to cumulative interference. This motivates memory updates that actively correct the response at the current key address rather than merely accumulating on top of the state.

2.2 DeltaNet: Error-Corrective Update and Resulting Past-Key Interference

DeltaNet (Schlag et al., 2021) mitigates the current-key mismatch of plain additive accumulation using an error-corrective delta-rule update. Instead of accumulating the outer product $\boldsymbol{v}_t\boldsymbol{k}_t^\top$, DeltaNet first evaluates the existing memory response at the current key. Let $\boldsymbol{r}_t^{\text{old}} := \boldsymbol{S}_{t-1}\boldsymbol{k}_t$ denote this response, and define the current-key prediction error as $\boldsymbol{e}_t := \boldsymbol{r}_t^{\text{old}} - \boldsymbol{v}_t$. DeltaNet then applies the rank-one update:

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta_t\boldsymbol{e}_t\boldsymbol{k}_t^\top = \boldsymbol{S}_{t-1} - \beta_t(\boldsymbol{r}_t^{\text{old}} - \boldsymbol{v}_t)\boldsymbol{k}_t^\top, \tag{3}$$

where $\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$ is the memory readout and β_t is an input-dependent write strength. By expanding equation 3 as $\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta_t(\boldsymbol{S}_{t-1}\boldsymbol{k}_t)\boldsymbol{k}_t^\top + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$, it becomes clear that DeltaNet removes a fraction of the existing response at the current key and adds the target value.

To examine its effect on memory responses, let $\boldsymbol{r}_j^{\text{new}} := \boldsymbol{S}_t\boldsymbol{k}_j$ and $\boldsymbol{r}_j^{\text{old}} := \boldsymbol{S}_{t-1}\boldsymbol{k}_j$. Applying equation 3 to any key $\boldsymbol{k}_j$ yields:

$$\boldsymbol{r}_j^{\text{new}} = \boldsymbol{r}_j^{\text{old}} - \beta_t\boldsymbol{e}_t(\boldsymbol{k}_t^\top\boldsymbol{k}_j) = \boldsymbol{r}_j^{\text{old}} - \beta_t(\boldsymbol{k}_t^\top\boldsymbol{k}_j)\left(\boldsymbol{r}_t^{\text{old}} - \boldsymbol{v}_t\right). \tag{4}$$

Since $\boldsymbol{k}_t^\top\boldsymbol{k}_t = 1$, setting $j = t$ yields $\boldsymbol{r}_t^{\text{new}} = (1-\beta_t)\boldsymbol{r}_t^{\text{old}} + \beta_t\boldsymbol{v}_t$ for the current key. Thus, DeltaNet moves the current-key response toward the target value, allowing the new association to replace the old response rather than merely accumulate on top of it. This replacement behavior is particularly effective in recency-sensitive associative retrieval, where the model must return the most recently assigned value for a queried key.

However, for any past key $j < t$, equation 4 reveals an unintended interference. This interference occurs because the correction signal leaks into any past key $\boldsymbol{k}_j$ that is not orthogonal to $\boldsymbol{k}_t$; the magnitude of this leakage is proportional to the key similarity $\boldsymbol{k}_t^\top\boldsymbol{k}_j$. Therefore, while DeltaNet successfully corrects the current-key prediction error, it fails to preserve the integrity of past-key responses. This observation suggests that the limitation is not the error-correction principle itself, but the lack of an interference-aware write direction that can be decoupled from the current key.

3 Interference-Aware DeltaNet

Building upon the analysis of DeltaNet's limitations, we propose Interference-Aware DeltaNet (IA-DeltaNet). This approach explicitly decouples the key used for error evaluation from the direction used to update the memory.

3.1 Main Idea

The core design of IA-DeltaNet is to maintain the current key $\boldsymbol{k}_t$ for measuring the prediction error, $\boldsymbol{e}_t = \boldsymbol{S}_{t-1}\boldsymbol{k}_t - \boldsymbol{v}_t$, while replacing the DeltaNet write direction $\boldsymbol{k}_t$ with an optimized interference-

aware direction $\boldsymbol{u}_t$. The resulting update rule is defined as:

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta_t \boldsymbol{e}_t \boldsymbol{u}_t^\top, \qquad \boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t, \tag{5}$$

where the correction signal $\boldsymbol{e}_t$, the write strength β_t, and the readout rule remain identical to those in DeltaNet. By changing only the direction along which the correction is written, we gain a degree of freedom to minimize memory disturbance.

Applying this update to a past key $\boldsymbol{k}_j$ ($j < t$) illustrates the impact on previous memory states:

$$\boldsymbol{S}_t \boldsymbol{k}_j = \boldsymbol{S}_{t-1} \boldsymbol{k}_j - \beta_t \boldsymbol{e}_t (\boldsymbol{u}_t^\top \boldsymbol{k}_j). \tag{6}$$

To preserve past associations while correcting the current one, the write direction $\boldsymbol{u}_t$ should satisfy two requirements: 1) it should minimize the disturbance to past-key responses, ensuring $\boldsymbol{S}_t \boldsymbol{k}_j \simeq \boldsymbol{S}_{t-1} \boldsymbol{k}_j$ for all $j < t$, and 2) it must preserve the same error-correction on the current key $\boldsymbol{k}_t$ as DeltaNet.

3.2 Objective Function and its Closed-Form Solution

To derive the optimal write direction $\boldsymbol{u}_t$, we first formulate the requirements introduced above using a constrained optimization framework.

Past-key interference objective. From equation 6, the update to the current memory state induces a change in the response of every past key $\boldsymbol{k}_j$. We quantify the total interference as the sum of the squared ℓ_2 norms of these interferences:

$$\mathcal{I}_t(\boldsymbol{u}_t) = \sum_{j=1}^{t-1} \left\| -\beta_t \boldsymbol{e}_t (\boldsymbol{u}_t^\top \boldsymbol{k}_j) \right\|_2^2 = \beta_t^2 \|\boldsymbol{e}_t\|_2^2 \sum_{j=1}^{t-1} (\boldsymbol{u}_t^\top \boldsymbol{k}_j)^2 = \beta_t^2 \|\boldsymbol{e}_t\|_2^2 \boldsymbol{u}_t^\top \left(\sum_{j=1}^{t-1} \boldsymbol{k}_j \boldsymbol{k}_j^\top \right) \boldsymbol{u}_t. \tag{7}$$

Since the write strength β_t and the error signal $\boldsymbol{e}_t$ are independent of the choice of $\boldsymbol{u}_t$, minimizing the total interference is equivalent to minimizing the quadratic form, $\boldsymbol{u}_t^\top \boldsymbol{\Sigma}_{t-1} \boldsymbol{u}_t$. Here, $\boldsymbol{\Sigma}_{t-1} := \sum_{j=1}^{t-1} \boldsymbol{k}_j \boldsymbol{k}_j^\top$ represents the uncentered covariance matrix of past keys, with $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$ ensuring numerical stability.

Current-key correction constraint. While minimizing interference, we must also ensure that the update effectively corrects the response at the current key $\boldsymbol{k}_t$. In DeltaNet, this correction is achieved because $\boldsymbol{k}_t^\top \boldsymbol{k}_t = 1$. To preserve exactly the same correction behavior in IA-DeltaNet, we impose the following linear constraint:

$$\boldsymbol{u}_t^\top \boldsymbol{k}_t = 1. \tag{8}$$

This constraint preserves the error-correction performance of DeltaNet while allowing $\boldsymbol{u}_t$ the flexibility to deviate from $\boldsymbol{k}_t$ to minimize interference with past-key responses.

Optimization. Combining the objective and the constraint, we obtain the following quadratic programming problem:

$$\boldsymbol{u}_t = \arg\min_{\hat{\boldsymbol{u}}_t} \ \hat{\boldsymbol{u}}_t^\top \boldsymbol{\Sigma}_{t-1} \hat{\boldsymbol{u}}_t \quad \text{s.t.} \quad \hat{\boldsymbol{u}}_t^\top \boldsymbol{k}_t = 1. \tag{9}$$

Using a Lagrange multiplier η, the Lagrangian is defined as

$$\mathcal{L}(\hat{\boldsymbol{u}}_t, \eta) = \hat{\boldsymbol{u}}_t^\top \boldsymbol{\Sigma}_{t-1} \hat{\boldsymbol{u}}_t - \eta (\hat{\boldsymbol{u}}_t^\top \boldsymbol{k}_t - 1). \tag{10}$$

The closed-form solution of this problem, representing the interference-aware write direction, is:

$$\boldsymbol{u}_t = \frac{\boldsymbol{\Sigma}_{t-1}^{-1} \boldsymbol{k}_t}{\boldsymbol{k}_t^\top \boldsymbol{\Sigma}_{t-1}^{-1} \boldsymbol{k}_t}. \tag{11}$$

The derived direction $\boldsymbol{u}_t$ can be interpreted as a preconditioned key. Specifically, $\boldsymbol{\Sigma}_{t-1}^{-1}$ acts as a whitening transform that downweights directions already heavily occupied by past keys, while the denominator renormalizes the vector to satisfy the current-key correction constraint.

IA-DeltaNet differs from DeltaNet when the past-key geometry is anisotropic; in the isotropic case ($\boldsymbol{\Sigma}_{t-1} \propto \boldsymbol{I}$), the optimal write direction reduces to $\boldsymbol{u}_t = \boldsymbol{k}_t$, recovering DeltaNet. Thus, IA-DeltaNet generalizes DeltaNet by adapting the write direction to how past keys are distributed across the representation space.

3.3 Geometric Interpretation of Interference Reduction

Although $|\boldsymbol{u}_t^\top \boldsymbol{k}_j| < |\boldsymbol{k}_t^\top \boldsymbol{k}_j|$ does not hold for every individual past key, IA-DeltaNet is mathematically guaranteed to reduce the total interference. Let $\mathcal{I}_t(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{\Sigma}_{t-1} \boldsymbol{w}$ be the total squared interference. For DeltaNet ($\boldsymbol{u}_t = \boldsymbol{k}_t$), the interference is $\boldsymbol{k}_t^\top \boldsymbol{\Sigma}_{t-1} \boldsymbol{k}_t$. For IA-DeltaNet, substituting the optimal solution from equation 11 yields:

$$\mathcal{I}_t(\boldsymbol{u}_t) = \frac{1}{\boldsymbol{k}_t^\top \boldsymbol{\Sigma}_{t-1}^{-1} \boldsymbol{k}_t}. \tag{12}$$

By applying the matrix-form Cauchy-Schwarz inequality, specifically $(\boldsymbol{k}_t^\top \boldsymbol{\Sigma}_{t-1} \boldsymbol{k}_t)(\boldsymbol{k}_t^\top \boldsymbol{\Sigma}_{t-1}^{-1} \boldsymbol{k}_t) \geq (\boldsymbol{k}_t^\top \boldsymbol{k}_t)^2 = 1$, we have the following inequality:

$$\mathcal{I}_t(\boldsymbol{u}_t) \leq \mathcal{I}_t(\boldsymbol{k}_t), \tag{13}$$

where the equality holds when $\boldsymbol{k}_t$ is an eigenvector of $\boldsymbol{\Sigma}_{t-1}$, in which case $\boldsymbol{u}_t = \boldsymbol{k}_t$ and IA-DeltaNet reduces to DeltaNet. In all other cases, IA-DeltaNet strictly reduces the total memory disturbance by "whitening" the write direction through $\boldsymbol{\Sigma}_{t-1}^{-1}$, effectively steering the update away from high-density regions of the past-key geometry.

3.4 Efficient Diagonal Approximation

While the exact solution in equation 11 provides the optimal interference-aware direction, it poses computational challenges for high-dimensional states. A direct implementation requires either a full matrix inversion or a linear system solve at each step. Although the inverse covariance can be updated via the Sherman–Morrison formula by exploiting the recursive structure, $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t-1} + \boldsymbol{k}_t \boldsymbol{k}_t^\top$, this approach still incurs an $O(d_k^2)$ cost per step. To maintain the linear-time efficiency characteristic of modern recurrent models, we adopt a diagonal approximation of the covariance matrix.

Specifically, we approximate the uncentered covariance matrix $\boldsymbol{\Sigma}_{t-1}$ with a diagonal matrix:

$$\boldsymbol{\Sigma}_{t-1} \approx \boldsymbol{D}_{t-1} = \mathrm{diag}(\boldsymbol{d}_{t-1}), \tag{14}$$

where the diagonal state $\boldsymbol{d}_t \in \mathbb{R}^{d_k}$ is computed as a prefix sum of elementwise squared keys:

$$\boldsymbol{d}_t = \boldsymbol{d}_{t-1} + \boldsymbol{k}_t \odot \boldsymbol{k}_t, \quad \boldsymbol{d}_0 = \boldsymbol{1}. \tag{15}$$

Substituting $\boldsymbol{D}_{t-1}$ into the solution in equation 11 yields the interference-aware write direction:

$$\boldsymbol{u}_t = \frac{\boldsymbol{D}_{t-1}^{-1} \boldsymbol{k}_t}{\boldsymbol{k}_t^\top \boldsymbol{D}_{t-1}^{-1} \boldsymbol{k}_t}. \tag{16}$$

In this formulation, the diagonal state $\boldsymbol{d}_{t-1}$ serves as an accumulator of feature-wise energy from past keys. Consequently, this approximation selectively downweights dimensions that have been heavily occupied by previous keys, then renormalizes the direction to ensure that the current-key correction constraint is strictly satisfied.

The computational overhead relative to the standard DeltaNet is minimal, consisting only of the element-wise construction of $\boldsymbol{u}_t$ and the update of $\boldsymbol{d}_t$. Crucially, the dominant operation remains the $O(d_v d_k)$ rank-one memory update; since the additional overhead for computing $\boldsymbol{u}_t$ is only $O(d_k)$, IA-DeltaNet maintains the same asymptotic complexity as the original DeltaNet. Moreover, since $\boldsymbol{d}_t$ can be computed by a simple prefix sum, the diagonal write directions can be computed efficiently in a chunk-parallel fashion using associative scans, preserving the efficiency of linear attention.

4 Experiments

Section 4.1 first evaluates IA-DeltaNet on associative recall tasks, which test whether stored associations are preserved. Sections 4.2 and 4.3 then evaluate long-document language modeling and streaming video question answering, and Section 4.4 concludes with analyses. Unless otherwise specified, IA-DeltaNet refers to the diagonal approximation described in Section 3.4, and exact IA-DeltaNet, marked with † in tables and figures, uses the exact write direction in equation 11.

Table 1: MQAR accuracy (%), mean$_{\pm\text{std}}$ over three seeds, with K key–value pairs and sequence length L. IA-DeltaNet† uses the exact write direction computed from the full past-key covariance.

d_{model}	K	L	DeltaNet	Gated DeltaNet	**IA-DeltaNet**	**IA-DeltaNet†**
64	32	256	$99.1_{\pm 0.1}$	$96.6_{\pm 3.2}$	$99.1_{\pm 0.2}$	$\mathbf{99.2}_{\pm 0.1}$
	64	512	$94.2_{\pm 2.0}$	$92.6_{\pm 8.0}$	$95.1_{\pm 0.9}$	$\mathbf{98.2}_{\pm 0.5}$
	128	1024	$50.7_{\pm 23.8}$	$26.2_{\pm 23.3}$	$64.0_{\pm 0.5}$	$\mathbf{70.0}_{\pm 5.7}$
128	64	512	$99.3_{\pm 0.2}$	$66.2_{\pm 57.3}$	$99.2_{\pm 0.1}$	$\mathbf{99.4}_{\pm 0.3}$
	128	1024	$99.1_{\pm 0.1}$	$32.8_{\pm 56.8}$	$99.1_{\pm 0.1}$	$\mathbf{99.6}_{\pm 0.2}$
	256	2048	$81.7_{\pm 14.9}$	$0.0_{\pm 0.0}$	$95.5_{\pm 1.1}$	$\mathbf{97.5}_{\pm 2.0}$

Table 2: S-NIAH accuracy (%). The 4K and 8K settings exceed the 2K training context.

Task	Model	512	1K	2K	4K	8K
S-NIAH-1	DeltaNet	100.0	100.0	100.0	100.0	99.8
	Gated DeltaNet	100.0	100.0	100.0	100.0	99.6
	IA-DeltaNet	100.0	100.0	100.0	100.0	**100.0**
S-NIAH-2	DeltaNet	100.0	100.0	60.2	32.2	10.0
	Gated DeltaNet	100.0	100.0	99.6	33.8	11.2
	IA-DeltaNet	100.0	100.0	**100.0**	**39.6**	**12.8**
S-NIAH-3	DeltaNet	96.0	74.4	38.0	15.2	5.6
	Gated DeltaNet	96.0	90.8	**59.6**	17.8	5.0
	IA-DeltaNet	**97.8**	**95.6**	49.4	**19.4**	**6.2**

4.1 Associative Recall

Associative recall requires storing key–value associations in the recurrent state and retrieving a value from its key after many subsequent writes. DeltaNet's past-key interference is most consequential here, since equation 4 shows that each subsequent write perturbs the stored responses in proportion to key overlap, and these perturbations accumulate as more associations are stored or as more writes follow them. We therefore vary how much information must be retained: the number of key–value pairs in MQAR, the context length in S-NIAH, and the number of LLM-visible frames in VNBench.

4.1.1 Synthetic Sequences

Multi-query associative recall (MQAR) (Arora et al., 2024) presents a sequence of key–value pairs and queries the values of past keys. We vary associative load through the number of key–value pairs K at model widths 64 and 128, with sequence length $L = 8 \times K$, and report results over three seeds.

Table 1 shows that the benefit of interference-aware writing grows with associative load. At low load, DeltaNet and IA-DeltaNet are both near saturation. At the highest load for each model width, IA-DeltaNet improves over DeltaNet from 50.7% to 64.0% at $d_{\text{model}} = 64$, $K = 128$ and from 81.7% to 95.5% at $d_{\text{model}} = 128$, $K = 256$, while substantially reducing seed-to-seed variation. Exact IA-DeltaNet improves further in every setting, reaching 70.0% and 97.5% at these loads, which shows the benefit of capturing the off-diagonal structure of the past-key covariance.

Furthermore, since interference-aware writing applies to the delta-rule update rather than to a specific architecture, it can also be combined with other recurrent memories that use this update. As an example, applying it to Gated DeltaNet (Yang et al., 2025) improves over the base model in every evaluated setting, as shown in Appendix A.2. The gains are largest at high load, where Gated DeltaNet degrades sharply, with accuracy rising from 32.8% to 98.8% at $d_{\text{model}} = 128$, $K = 128$.

4.1.2 Language Models

S-NIAH (Hsieh et al., 2024) inserts a key–value pair into a long distractor context and queries the value by its key. We train language models with DeltaNet, Gated DeltaNet, or IA-DeltaNet layers from scratch on the same 15.7B-token SlimPajama (Soboleva et al., 2023) stream with a 2K context length, following Yang et al. (2024), and evaluate them on contexts from 512 to 8K tokens.

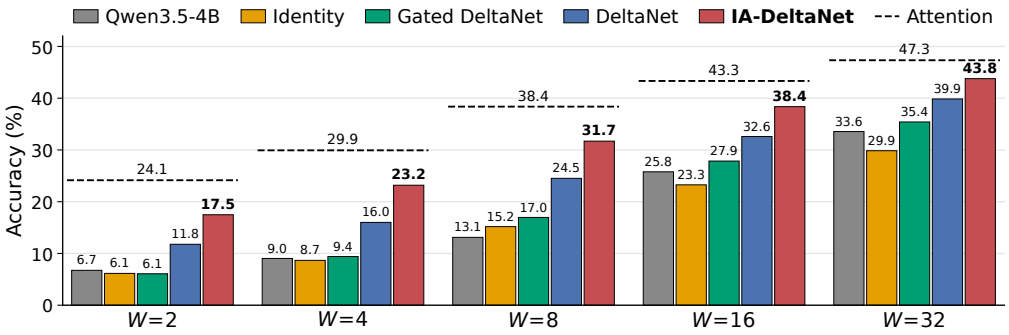


Figure 1: VNBench accuracy across LLM-visible window sizes W. Dashed lines indicate attention, which retains all past KVs. IA-DeltaNet outperforms recurrent memory baselines at every W.

As shown in Table 2, recall degrades as longer contexts place more distractor writes into the fixed-size state, and interference-aware writing slows this degradation. IA-DeltaNet matches or exceeds DeltaNet in every setting, with the largest differences on S-NIAH-2 and S-NIAH-3. Beyond the 2K training context, IA-DeltaNet achieves the highest accuracy among all models in every setting where not all models saturate.

4.1.3 Streaming Video

VNBench (Zhao et al., 2025b) inserts synthetic needle frames into long videos and queries them. We evaluate it under a streaming protocol built on Qwen3.5-4B (QwenTeam, 2026), inserting two memory blocks between the vision aligner and the LLM, each using DeltaNet, Gated DeltaNet, or IA-DeltaNet as its temporal mixer. The memory blocks process the entire visual stream in temporal order, whereas the LLM sees only the readouts of the last W frames, so earlier information must be carried by the recurrent state. We compare them with two no-memory baselines, Qwen3.5-4B and Identity, which removes the temporal mixer, and with a full-KV reference that uses attention over all past frames. All fine-tuned models are trained with $W = 8$ and evaluated at $W \in \{2, 4, 8, 16, 32\}$.

As shown in Figure 1, IA-DeltaNet outperforms both DeltaNet and Gated DeltaNet at every W. At $W = 2$, where the LLM must rely almost entirely on the recurrent state, IA-DeltaNet improves accuracy over DeltaNet from 11.8% to 17.5%. At larger W, it narrows the gap to attention, trailing by only 3.5 points at $W = 32$ while maintaining a fixed-size state.

Taken together, interference-aware writing consistently improves associative recall across all three settings, most when the fixed-size state must retain more information. Experimental details for the three settings are provided in Appendices A, B, and C.

4.2 Long-Document Language Modeling

Using the same language models as in Section 4.1.2, we evaluate perplexity on PG-19 (Rae et al., 2020) documents of up to 128K tokens under document-level state carry. Each document is processed in consecutive 2K-token chunks, with recurrent states propagated across chunk boundaries and reset only at document boundaries, which tests whether the state remains reliable far beyond the 2K training context.

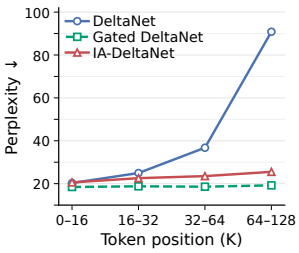


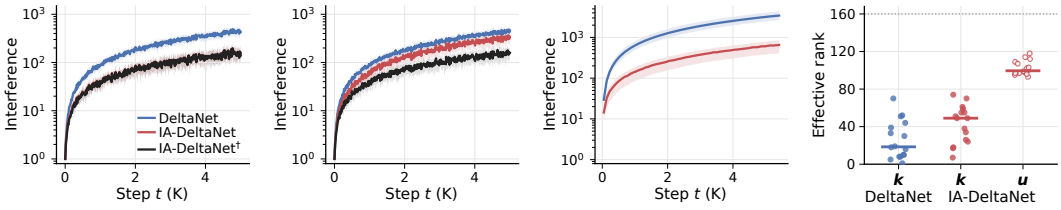
Figure 2: PG-19 perplexity.

As shown in Figure 2, DeltaNet and IA-DeltaNet achieve similar perplexity over the first 16K tokens but diverge as the state is propagated, with DeltaNet rising from 20.30 to 90.84 at 64–128K and IA-DeltaNet only from 20.60 to 25.51. This suggests that interference accumulated over many writes corrupts DeltaNet's state, which interference-aware writing largely prevents.

Gated DeltaNet remains the most stable throughout, as its forget gate prevents such accumulation by decaying past content, at little cost to perplexity, which depends mostly on nearby tokens (Khandelwal et al., 2018; Sun et al., 2021). This forgetting, however, also discards stored associations that recall requires, whereas interference-aware writing retains them while limiting the interference among them. The two mechanisms therefore act on different parts of the update and can be combined, as applying interference-aware writing to Gated DeltaNet improves its associative recall in Section 4.1.1.

Table 3: Streaming video QA accuracy (%) with the last W frames visible to the LLM. Bold indicates the best result among fixed-size memories; ‡ denotes the full-KV reference.

Model	Memory	TempCompass			NExT-QA			MVBench			TemporalBench		
		$W=2$	4	8	$W=2$	4	8	$W=2$	4	8	$W=2$	4	8
Qwen3.5-4B	None	54.4	60.0	66.3	67.4	70.2	72.7	51.6	57.0	58.7	55.0	56.9	58.0
+ Identity		59.5	62.9	64.9	66.4	69.1	71.9	51.9	54.5	55.9	61.4	63.1	64.2
+ DeltaNet		59.4	62.6	66.4	60.0	66.2	73.5	**55.2**	57.7	58.3	62.0	64.5	64.6
+ Gated DeltaNet	$\mathcal{O}(1)$	58.6	62.7	67.5	67.5	70.0	73.6	52.4	56.3	58.1	61.1	64.2	**65.4**
+ **IA-DeltaNet**		**65.4**	**67.7**	**70.1**	**71.1**	**75.0**	**76.9**	**55.2**	**58.4**	**59.2**	**63.5**	**64.6**	**65.4**
+ Attention‡	$\mathcal{O}(N)$	63.2	66.8	68.8	71.7	75.4	76.7	56.7	57.6	58.1	64.4	65.8	65.7



(a) Synthetic keys (b) Rotated synthetic keys (c) Real video keys (d) Effective rank

Figure 3: Write interference measured on (a) synthetic keys, (b) rotated synthetic keys, and (c) real video keys. (d) Effective rank of keys ($\boldsymbol{k}$) and IA-DeltaNet write directions ($\boldsymbol{u}$) on real video keys.

4.3 Streaming Video Question Answering

We next test whether the recall gains on VNBench transfer to downstream video QA under the same setup. We evaluate TempCompass (Liu et al., 2024), NExT-QA (Xiao et al., 2021), MVBench (Li et al., 2024a), and TemporalBench (Cai et al., 2024), selected because their accuracy improves with W, indicating reliance on information beyond a short local window, as detailed in Appendix C.2.

Table 4: Video-MME results.

Model	Video-MME
Qwen3.5-4B	49.7
+ Identity	50.4
+ DeltaNet	53.5
+ Gated DeltaNet	51.4
+ **IA-DeltaNet**	**54.9**
+ Attention‡	58.4

As shown in Table 3, IA-DeltaNet achieves the best performance among fixed-size memories in every setting. The gains over DeltaNet are most pronounced on TempCompass and NExT-QA, reaching 6.0 and 11.1 points at $W=2$, which indicates that the recall gains carry over to temporal reasoning. Despite its fixed-size state, IA-DeltaNet also remains competitive with attention.

We also evaluate Video-MME (Fu et al., 2025), which covers videos of diverse lengths, using $W=8$, 64, and 256 for its short, medium, and long subsets. As shown in Table 4, IA-DeltaNet performs best among fixed-size memories, while its gap to attention is larger than on the shorter videos in Table 3.

4.4 Analysis

We analyze the memory writes directly through the write interference $\mathcal{I}_t(\boldsymbol{w})=\boldsymbol{w}^\top\boldsymbol{\Sigma}_{t-1}\boldsymbol{w}$ defined in Section 3.3, which measures the total squared overlap of a write direction $\boldsymbol{w}$ with past keys.

Synthetic keys. To test how closely the diagonal approximation follows the exact write direction, we draw unit-norm synthetic keys from a Gaussian with an anisotropic spectrum, whose covariance is diagonal when its eigenbasis is aligned with the coordinate axes and gains off-diagonal structure when the eigenbasis is rotated. As shown in Figure 3a, IA-DeltaNet nearly matches exact IA-DeltaNet under a diagonal covariance, and both incur much lower interference than DeltaNet. Because the diagonal approximation cannot capture off-diagonal structure, a rotated covariance places IA-DeltaNet between DeltaNet and exact IA-DeltaNet, as shown in Figure 3b. This is consistent with the further gains of exact IA-DeltaNet on MQAR in Table 1. The full sweep is shown in Appendix D.

Real video keys. As shown in Figure 3c, on keys that the trained video models produce from real videos, IA-DeltaNet incurs lower interference than DeltaNet throughout the stream, and the gap widens as more keys are written, confirming that the reduction carries over to learned keys. Figure 3d further shows that IA-DeltaNet writes along a broader subspace, as its write directions reach a median effective rank of 100 out of $d_k = 160$, compared with 49 for its keys and 19 for DeltaNet's keys.

5 Related Work

Fixed-size recurrent memory. Softmax attention (Vaswani et al., 2017) stores all past keys and values, so its memory and compute grow with the sequence length. Linear attention (Katharopoulos et al., 2020) instead compresses the past into a fixed-size matrix-valued state updated recurrently, closely related to state space models (Gu & Dao, 2023; Dao & Gu, 2024; Lahoti et al., 2026), and test-time learning approaches further expand this state through learned memory updates (Sun et al., 2025; Behrouz et al., 2025). DeltaNet replaces additive accumulation with an error-corrective delta-rule update (Schlag et al., 2021), and subsequent work has developed chunk-parallel training (Yang et al., 2024), gated forgetting at scalar or channel-wise granularity (Yang et al., 2025; Team et al., 2025; Hatamizadeh et al., 2026), and richer state transitions with separate removal keys (Peng et al., 2025) or multiple delta-rule steps per token (Siems et al., 2025). IA-DeltaNet instead targets the write direction, deriving it from past-key interference rather than from the current token alone, which makes it complementary to gated forgetting.

Associative memory as regression. Another line of work views associative memory as regression over stored key–value associations (Wang et al., 2026). MesaNet (von Oswald et al., 2026) maintains decayed accumulators of key–value products and the key Gram matrix instead of a delta-rule state, and solves this regression at every read by running conjugate-gradient iterations for each query. Recent work keeps the delta rule and preconditions its update to better approximate this solution, either from the curvature of online least squares (Tumma et al., 2026) or through a feature-wise scaling adapted online by hypergradient feedback (Zhou et al., 2026). IA-DeltaNet instead starts from the interference that a write induces on stored associations and derives its write direction in closed form under the constraint that DeltaNet's current-key correction is exactly preserved, without additional trainable parameters.

Memory-efficient long-context understanding. A broad line of work addresses long-context modeling by retaining, compressing, or retrieving a bounded subset of past information. StreamingLLM (Xiao et al., 2024) maintains attention sinks and recent tokens for stable streaming inference, while H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024b), and PyramidKV (Cai et al., 2025) reduce the KV cache by selecting important past tokens under a fixed memory budget. Long-form and streaming VideoLLMs similarly reduce memory pressure through frame sampling, token compression, memory banks, sparse long-term memories, or video-specific KV-cache mechanisms (Lin et al., 2024; Chen et al., 2025; He et al., 2024; Song et al., 2024; Zhang et al., 2025a; Qian et al., 2024; Di et al., 2025). These approaches primarily manage which past representations are retained, compressed, or retrieved, whereas IA-DeltaNet studies how new information is written into a fixed-size recurrent associative state.

6 Conclusion

We presented IA-DeltaNet, which decouples the key used for error evaluation from the direction along which the delta-rule correction is written. Its write direction minimizes interference with past keys while exactly preserving DeltaNet's current-key correction, and a diagonal approximation retains linear-time efficiency and chunk-parallel training. IA-DeltaNet improves associative recall in synthetic, language, and video settings, most when the fixed-size state must retain more information, carries these gains over to streaming video question answering, and substantially mitigates DeltaNet's degradation under prolonged state propagation. Since it applies to the delta-rule update itself, it also improves Gated DeltaNet on associative recall, complementing gated forgetting. Reduced write interference and broader write directions accompany these gains, suggesting that accounting for past-key geometry is an effective principle for associative memory in linear-attention architectures.

AI USE STATEMENT

In this work, we used generative AI tools for method implementation, result interpretation, and feedback on research methodology and experimental design. We have not used generative AI tools for the theoretical formulation, mathematical derivations, hypotheses, synthetic data generation, or translation, and dataset cleaning or reformatting and qualitative data analyses are not applicable to this work. Additionally, we used generative AI tools for literature search, refinement of author-written text, and figure creation. We have reviewed all AI-assisted work. AI-assisted code was tested against reference implementations, and AI-assisted writing was reviewed line by line. We also consulted original sources before citing them. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, and all experiments use publicly available models and datasets in accordance with their licenses; we release no new datasets. IA-DeltaNet is a general-purpose memory update rule, so its societal impact depends on the downstream application.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the model configurations, training details, and evaluation protocols for associative recall (Appendix A), language modeling (Appendix B), and streaming video (Appendix C), as well as the synthetic analysis setup (Appendix D) and the algorithms (Appendix E).

REFERENCES

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Y Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *ICLR*, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *NeurIPS*, 2025.

Mu Cai, Reuben Tan, Jianrui Zhang, Bocheng Zou, Kai Zhang, Feng Yao, Fangrui Zhu, Jing Gu, Yiwu Zhong, Yuzhang Shang, et al. TemporalBench: Benchmarking fine-grained temporal understanding for multimodal video models. In *NeurIPS, workshop on Video-Language Models*, 2024.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *CoLM*, 2025.

Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Yihui He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Linxi Fan, Yuke Zhu, Yao Lu, and Song Han. LongVILA: Scaling long-context visual language models for long videos. In *ICLR*, 2025.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *ICML*, 2024.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, Hao Jiang, et al. Streaming video question-answering with in-context video KV-cache retrieval. In *ICLR*, 2025.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *CVPR*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv*, 2023.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: Decoupling erase and write in linear attention. *arXiv*, 2026.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. MA-LMM: Memory-augmented large multimodal model for long-term video understanding. In *CVPR*, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *CoLM*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*, 2018.

Aakash Sunil Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *ICLR*, 2026.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. MVBench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, 2024a.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. *NeurIPS*, 2024b.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. In *EMNLP*, 2024.

Junming Lin, Zheng Fang, Chi Chen, Haoxuan Cheng, Zihao Wan, Fuwen Luo, Ziyue Wang, Peng Li, Yang Liu, and Maosong Sun. StreamingBench: Assessing the gap for MLLMs to achieve streaming video understanding. In *ICASSP*, 2026.

Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. TempCompass: Do video LLMs really understand videos? In *ACL Findings*, 2024.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, et al. OVO-Bench: How far is your video-LLMs from real-world online video understanding? In *CVPR*, 2025.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with expressive dynamic state evolution. In *CoLM*, 2025.

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. *NeurIPS*, 2024.

QwenTeam. Qwen3.5: Towards native multimodal agents, 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *ICML*, 2021.

Yujiao Shen, Shulin Tian, Jingkang Yang, and Ziwei Liu. A simple baseline for streaming video understanding. *arXiv*, 2026.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via Householder products. *NeurIPS*, 2025.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. Slimpajama: A 627b token cleaned and deduplicated version of redpajama, 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *CVPR*, 2024.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. Do long-range language models actually use long-range context? In *EMNLP*, 2021.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *ICML*, 2025.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv*, 2025.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned DeltaNet: Curvature-aware sequence modeling for linear recurrences. In *ICML*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS*, 2017.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, et al. Mesanet: Sequence modeling by locally optimal test-time training. In *ICLR*, 2026.

Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *JMLR*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. NExT-QA: Next phase of question-answering to explaining temporal actions. In *CVPR*, 2021.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *NeurIPS*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *ICLR*, 2025.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. Flash-VStream: Efficient real-time understanding for long video streams. In *ICCV*, 2025a.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun MA, Ziwei Liu, and Chunyuan Li. LLaVA-Video: Video instruction tuning with synthetic data. *TMLR*, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *NeurIPS*, 2023.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, et al. Swift: a scalable lightweight infrastructure for fine-tuning. In *AAAI*, 2025a.

Zijia Zhao, Haoyu Lu, Yuqi Huo, Yifan Du, Tongtian Yue, Longteng Guo, Bingning Wang, Jing Liu, et al. Needle in a video haystack: A scalable synthetic evaluator for video MLLMs. In *ICLR*, 2025b.

Chenyu Zhou, Hongpei Li, Yuerou Liu, Jianghao Lin, Dongdong Ge, and Yinyu Ye. Osdn: Improving delta rule with provable online preconditioning in linear attention. *arXiv*, 2026.

A Multi-Query Associative Recall (MQAR)

A.1 Experimental Setup

We follow the MQAR task from Zoology (Arora et al., 2024) and set the sequence length to $L = 8 \times K$ for K key-value pairs. We evaluate model widths 64 and 128: for width 64, $K \in \{32, 64, 128\}$, and for width 128, $K \in \{64, 128, 256\}$. Each model contains two sequence-mixing layers with two heads and short convolutions of kernel size 4, interleaved with two MLP channel-mixing layers with expansion factor 4. We train each configuration with AdamW, weight decay 0.1, cosine learning-rate decay, and batch size 128 for up to 64 epochs, with early stopping once held-out accuracy exceeds 99%. We sweep learning rates over $\{10^{-4}, 3.2\times10^{-4}, 10^{-3}, 3.2\times10^{-3}\}$. Following Zoology, we report the best held-out accuracy over training epochs and learning rates for each method, configuration, and seed. Results are reported as the mean and standard deviation over three seeds.

A.2 Application to Gated DeltaNet

Since interference-aware writing applies to the delta-rule update rather than to a specific architecture, it can also be combined with other recurrent memories that use this update. We test this on Gated DeltaNet (Yang et al., 2025), which decays the previous state with a data-dependent gate $\alpha_t \in (0, 1]$ before applying the delta-rule correction:

$$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} - \beta_t \left(\alpha_t \boldsymbol{S}_{t-1} \boldsymbol{k}_t - \boldsymbol{v}_t\right) \boldsymbol{k}_t^\top. \tag{A.1}$$

As in DeltaNet, the correction is written along the current key $\boldsymbol{k}_t$. We keep the decay gate and the error term unchanged and replace only this write direction with the interference-aware direction $\boldsymbol{u}_t$ in equation 16:

$$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} - \beta_t \left(\alpha_t \boldsymbol{S}_{t-1} \boldsymbol{k}_t - \boldsymbol{v}_t\right) \boldsymbol{u}_t^\top. \tag{A.2}$$

We refer to this variant as IA-Gated DeltaNet. Its past-key statistic $\boldsymbol{d}_{t-1}$ is accumulated without the decay gate, as in IA-DeltaNet, and all other components, including projections, short convolutions, and output gating, follow the Gated DeltaNet configuration. Both models are trained and evaluated under the protocol in Appendix A.1.

Table A.1: MQAR accuracy (%) of Gated DeltaNet with and without the interference-aware write direction, mean$_{\pm\text{std}}$ over three seeds, with K key–value pairs and sequence length L.

d_{model}	K	L	Gated DeltaNet	**IA-Gated DeltaNet**
64	32	256	$96.6_{\pm3.2}$	$\mathbf{98.8}_{\pm0.3}$
	64	512	$92.6_{\pm8.0}$	$\mathbf{93.5}_{\pm3.1}$
	128	1024	$26.2_{\pm23.3}$	$\mathbf{74.7}_{\pm17.9}$
128	64	512	$66.2_{\pm57.3}$	$\mathbf{99.1}_{\pm0.1}$
	128	1024	$32.8_{\pm56.8}$	$\mathbf{98.8}_{\pm0.9}$
	256	2048	$0.0_{\pm0.0}$	$\mathbf{54.4}_{\pm47.7}$

Table A.1 compares Gated DeltaNet with and without interference-aware writing, and DeltaNet results are reported in Table 1. IA-Gated DeltaNet improves over Gated DeltaNet in all reported settings, with the largest gains at high load, where Gated DeltaNet degrades sharply. For example, accuracy rises from 26.2% to 74.7% at $d_{\text{model}} = 64$, $K = 128$ and from 32.8% to 98.8% at $d_{\text{model}} = 128$, $K = 128$. These results indicate that interference-aware writing and gated forgetting are complementary.

B Language Model Experiments

B.1 Models and Training

All models use 24 layers with model width 1,024, 8 attention heads, key and value dimensions of 128, and an MLP expansion factor of 4. DeltaNet and IA-DeltaNet each have approximately 374M trainable parameters, while Gated DeltaNet has approximately 400M. Unlike the other experiments, which follow Gated DeltaNet's default configuration with the value dimension doubled to $2d_v$, we keep it at d_v here to keep the parameter counts comparable. We train all models on the same

SlimPajama stream for 30K steps with a global batch size of 256, corresponding to 15.7B tokens. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.01, gradient clipping at 1.0, and a learning rate that linearly increases to 4×10^{-4} over 1K steps and then follows cosine decay to 4×10^{-5}. We use the same checkpoints for the S-NIAH and PG-19 evaluations, which differ only in their inference protocol and task.

B.2 S-NIAH Evaluation Details

In S-NIAH, each example inserts a needle pairing a word key with a value into a distractor context, and the model must generate the value given the key. S-NIAH-1 uses a numeric value in a haystack of repeated filler sentences, S-NIAH-2 uses a numeric value in a haystack of essays, and S-NIAH-3 uses a UUID value in a haystack of essays.

B.3 PG-19 Evaluation Details

For the PG-19 results in Section 4.2, we process all 100 test books (Rae et al., 2020) in consecutive 2K-token chunks, propagating the recurrent states across chunks and resetting them only at document boundaries. For each book, we evaluate up to the first 128K tokens and report token-weighted perplexity by document position. Unlike the standard PG-19 evaluation, which reports word-level perplexity, we report tokenizer-token perplexity under our document-level state-carry protocol.

C Streaming Video Understanding

C.1 Models and Training

Model configurations. Table C.1 summarizes the trainable parameters introduced by each memory block type. The LLM LoRA parameters are shared across all variants and are excluded from these counts. The identity baseline contains no temporal-mixing parameters, so its memory blocks consist only of the shared FFN and normalization layers. DeltaNet and IA-DeltaNet have the same parameter count because IA-DeltaNet changes only the write direction without introducing additional trainable parameters. Gated DeltaNet has more parameters because we follow its default configuration.

Table C.1: Trainable parameters per memory block for each temporal mixer. All video models use two memory blocks. The LLM LoRA parameters, which are shared across all variants, are excluded. Identity denotes the variant with no temporal-mixing layer.

Memory block	Parameters
Identity	53.1M
DeltaNet	79.4M
Gated DeltaNet	105.6M
IA-DeltaNet	79.4M

Training details. We use MS-SWIFT (Zhao et al., 2025a) to train the video models in Sections 4.1.3 and 4.3 on LLaVA-Video-178K (Zhang et al., 2025b) for one epoch on 16 H200 GPUs with an effective batch size of 64. We sample videos at 1 FPS, cap training clips at 180 frames, and process full videos during evaluation. The LLM's LoRA adapters use rank 8 and scaling factor $\alpha = 16$. We optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$), using learning rates of 2×10^{-4} for the memory blocks and 2×10^{-5} for the LoRA parameters, cosine decay schedule, and a 3% warmup ratio. We use a per-frame visual-token budget of 16 to 128 for both training and evaluation.

C.2 Benchmark Selection

Visible-window sensitivity analysis. We assess whether a benchmark is suitable for evaluating memory under our streaming protocol by measuring how accuracy changes with the number of LLM-visible frames W. If accuracy improves substantially with larger W, the benchmark is more likely to require information beyond a short local window. We sample each video at 1 FPS and provide only the last W frames to Qwen3.5-4B (QwenTeam, 2026); for videos longer than three

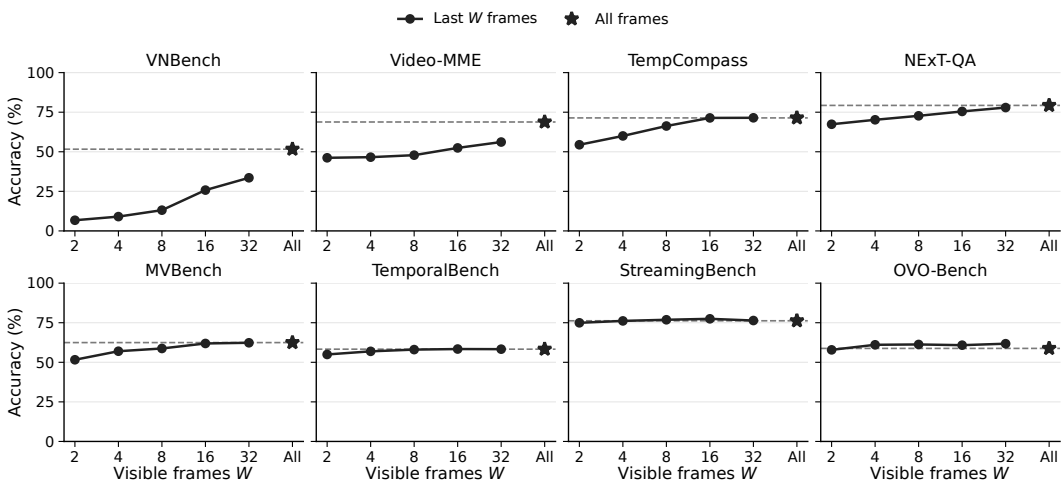


Figure C.1: Visible-window sensitivity of Qwen3.5. Benchmarks are sorted by the accuracy gain from $W = 2$ to all sampled frames. Benchmarks with larger gains are used in the main evaluation because they provide a clearer test of memory beyond a short visible window.

minutes, we uniformly sample 180 frames before selecting the last W sampled frames. Figure C.1 reports accuracy for $W \in \{2, 4, 8, 16, 32\}$ and for all sampled frames.

Video QA benchmarks. As shown in Figure C.1, the evaluated video QA benchmarks generally improve as the visible window W increases. VNBench (Zhao et al., 2025b) shows the largest gain as W increases, as expected from its needle-in-the-haystack design. Video-MME (Fu et al., 2025), TempCompass (Liu et al., 2024), NExT-QA (Xiao et al., 2021), TemporalBench (Cai et al., 2024), and MVBench (Li et al., 2024a) also generally improve with larger visible windows, indicating that they benefit from information beyond a short local window. Although these benchmarks were originally designed for offline video QA, their visible-window sensitivity supports their use for evaluating memory under our streaming protocol.

Streaming QA benchmarks. Streaming-oriented benchmarks such as StreamingBench (Lin et al., 2026) and OVO-Bench (Niu et al., 2025) follow a query-time evaluation protocol, where each question is answered using the video stream observed up to its query time rather than the entire video. Accordingly, we sample frames at 1 FPS up to each query time and provide only the last W frames to Qwen3.5. At $W = 2$, the drop relative to all sampled frames is only 1.3 points on StreamingBench and 0.9 points on OVO-Bench. At $W = 8$, using only the last 8 frames even outperforms all sampled frames on both StreamingBench (76.9% vs. 76.2%) and OVO-Bench (61.3% vs. 58.8%). This is consistent with SimpleStream (Shen et al., 2026), which similarly finds that recent-frame sliding-window baselines are highly competitive on OVO-Bench and StreamingBench. Together, these results suggest that a short recent window is already highly competitive on these benchmarks, making them less suitable for evaluating whether a recurrent state preserves information over the full stream. For this reason, we focus the main evaluation on benchmarks whose performance depends more clearly on information beyond a short local window.

C.3 Efficiency Measurement

Figure C.2 reports the per-step cost of the two-block memory module of our video models, where each step writes 91 new visual tokens after N tokens have been streamed. We measure it on an RTX A5000, using Triton kernels for DeltaNet and IA-DeltaNet and FlashAttention-2 for attention. DeltaNet and IA-DeltaNet keep a constant per-step cost regardless of stream length, and IA-DeltaNet adds only 0.4 ms of kernel time per step over DeltaNet. Attention instead attends to all past tokens, so its time and memory grow linearly with the stream, exceeding the per-step time of IA-DeltaNet beyond about 10K tokens and running out of memory at 1.5M tokens.

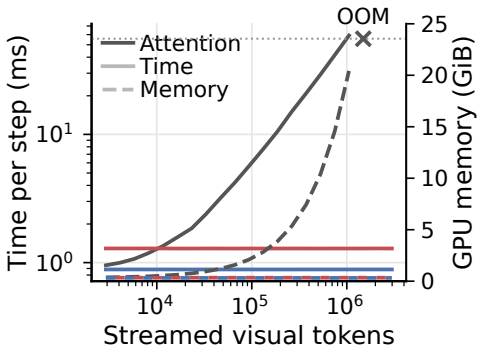


Figure C.2: Per-step GPU kernel time (solid) and peak memory (dashed) of the two-block memory module for DeltaNet (blue), IA-DeltaNet (red), and attention (gray) as the stream grows. $\times$ marks out-of-memory.

D Synthetic Interference Analysis

D.1 Synthetic Key Distribution

We construct synthetic keys with controlled covariance geometry to evaluate the diagonal approximation under varying anisotropy and off-diagonal structure. The construction separates the eigenvalue spectrum, which controls anisotropy, from the eigenvector basis, whose rotation controls off-diagonal structure of the covariance.

Specifically, we draw $\tilde{\boldsymbol{k}}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}(\theta))$ and normalize $\boldsymbol{k}_t = \tilde{\boldsymbol{k}}_t / \|\tilde{\boldsymbol{k}}_t\|_2$. We construct the covariance as $\boldsymbol{C}(\theta) = \boldsymbol{Q}(\theta)\boldsymbol{\Lambda}\boldsymbol{Q}(\theta)^\top$. Here, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_{d_k})$ sets the anisotropic eigenvalue spectrum, where $\lambda_i = \exp(-\gamma(i-1))$ for $i = 1, \ldots, d_k$. The decay rate γ controls the anisotropy of the key distribution, which we quantify by the participation ratio of the spectrum,

$$r_{\text{eff}} = \frac{\left(\sum_{i=1}^{d_k} \lambda_i\right)^2}{\sum_{i=1}^{d_k} \lambda_i^2}, \tag{D.1}$$

which is smaller when the keys concentrate in fewer directions. To vary the covariance orientation while preserving its eigenvalue spectrum, we sample a random matrix $\boldsymbol{R}$ with $R_{ij} \sim \mathcal{N}(0, 1)$ and construct

$$\boldsymbol{A} = \boldsymbol{R} - \boldsymbol{R}^\top, \qquad \boldsymbol{A} \leftarrow \frac{\sqrt{d_k}\boldsymbol{A}}{\|\boldsymbol{A}\|_F}. \tag{D.2}$$

We then set $\boldsymbol{Q}(\theta) = \exp(\theta \boldsymbol{A})$, which is orthogonal by construction since $\boldsymbol{A}$ is skew-symmetric ($\boldsymbol{A}^\top = -\boldsymbol{A}$). At $\theta = 0$, the covariance is diagonal; increasing θ rotates the eigenspace away from the coordinate axes, introducing off-diagonal structure. The same $\boldsymbol{A}$ is reused across all values of θ within each trial.

D.2 Interference Sweep

We evaluate the interference $\mathcal{I}_t(\boldsymbol{w})$ of DeltaNet ($\boldsymbol{w} = \boldsymbol{k}_t$), IA-DeltaNet ($\boldsymbol{w} = \boldsymbol{u}_t$ from the diagonal approximation), and exact IA-DeltaNet ($\boldsymbol{w} = \boldsymbol{u}_t$ from the full covariance) on these keys. We use $d_k = 160$, sequence length $T = 5000$, and average over 15 independent trials per setting. We sweep $\theta \in \{0, \pi/32, \pi/16, \pi/8, \pi/4\}$ and choose γ to obtain target participation ratios $r_{\text{eff}} \in \{5, 10, 20, 80, 160\}$. The diagonal and rotated covariances in Figure 3 correspond to $\theta = 0$ and $\theta = \pi/16$ with $r_{\text{eff}} = 10$.

Figure D.1 reports the full sweep. Under a diagonal covariance, IA-DeltaNet matches exact IA-DeltaNet at every anisotropy level. Rotation widens the gap between them, and the gap grows faster for more anisotropic spectra. Even under the strongest rotation, IA-DeltaNet incurs less interference than DeltaNet whenever the spectrum is anisotropic, and all three coincide as the spectrum approaches isotropy, where the interference-aware direction reduces to the key.

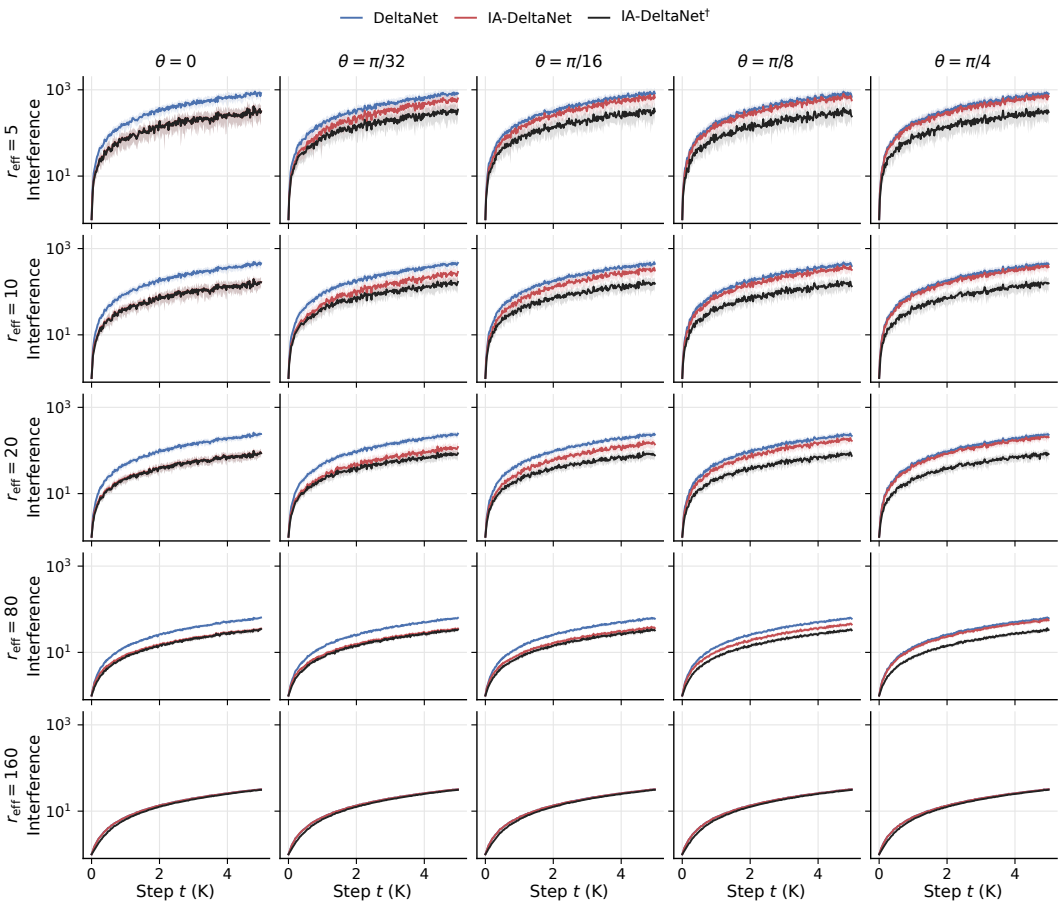


Figure D.1: Synthetic interference sweep over covariance rotation θ (columns) and participation ratio r_{eff} (rows).

E Algorithms and Implementation Details

E.1 Recurrent Updates

Diagonal IA-DeltaNet. Algorithm 1 summarizes the diagonal recurrent update described in Section 3.4.

Algorithm 1 Diagonal IA-DeltaNet in recurrent form.

Require: keys $\{\boldsymbol{k}_t\}_{t=1}^{T}$, queries $\{\boldsymbol{q}_t\}_{t=1}^{T}$, values $\{\boldsymbol{v}_t\}_{t=1}^{T}$, write strengths $\{\beta_t\}_{t=1}^{T}$

Initialize $\boldsymbol{S}_0 \leftarrow \boldsymbol{0} \in \mathbb{R}^{d_v \times d_k}$
Initialize $\boldsymbol{d}_0 \leftarrow \boldsymbol{1} \in \mathbb{R}^{d_k}$
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{D}_{t-1} \leftarrow \mathrm{diag}(\boldsymbol{d}_{t-1})$
  $\boldsymbol{u}_t \leftarrow \boldsymbol{D}_{t-1}^{-1}\boldsymbol{k}_t / (\boldsymbol{k}_t^{\top}\boldsymbol{D}_{t-1}^{-1}\boldsymbol{k}_t)$ ▷ IA write direction, equation 16
  $\boldsymbol{e}_t \leftarrow \boldsymbol{S}_{t-1}\boldsymbol{k}_t - \boldsymbol{v}_t$
  $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_{t-1} - \beta_t \boldsymbol{e}_t \boldsymbol{u}_t^{\top}$ ▷ IA-DeltaNet update, equation 5
  $\boldsymbol{o}_t \leftarrow \boldsymbol{S}_t \boldsymbol{q}_t$
  $\boldsymbol{d}_t \leftarrow \boldsymbol{d}_{t-1} + \boldsymbol{k}_t \odot \boldsymbol{k}_t$ ▷ Update diagonal state, equation 15
end for
return $\{\boldsymbol{o}_t\}_{t=1}^{T}$

Exact IA-DeltaNet. Algorithm 2 presents the recurrent form of exact IA-DeltaNet, which uses the full past-key covariance. Compared to DeltaNet, exact IA-DeltaNet computes the interference-aware write direction $\boldsymbol{u}_t$ from equation 11 and updates the past-key covariance $\boldsymbol{\Sigma}_t$. The memory update then replaces the write-side key $\boldsymbol{k}_t$ with $\boldsymbol{u}_t$.

Algorithm 2 Exact IA-DeltaNet in recurrent form.

Require: keys $\{\boldsymbol{k}_t\}_{t=1}^T$, queries $\{\boldsymbol{q}_t\}_{t=1}^T$, values $\{\boldsymbol{v}_t\}_{t=1}^T$, write strengths $\{\beta_t\}_{t=1}^T$

$\boldsymbol{S}_0 \leftarrow \boldsymbol{0} \in \mathbb{R}^{d_v \times d_k}$
$\boldsymbol{\Sigma}_0 \leftarrow \boldsymbol{I} \in \mathbb{R}^{d_k \times d_k}$
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{z}_t \leftarrow \boldsymbol{\Sigma}_{t-1}^{-1} \boldsymbol{k}_t$
  $\boldsymbol{u}_t \leftarrow \boldsymbol{z}_t / (\boldsymbol{k}_t^\top \boldsymbol{z}_t)$ ▷ IA write direction equation 11
  $\boldsymbol{e}_t \leftarrow \boldsymbol{S}_{t-1} \boldsymbol{k}_t - \boldsymbol{v}_t$
  $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_{t-1} - \beta_t \, \boldsymbol{e}_t \, \boldsymbol{u}_t^\top$
  $\boldsymbol{\Sigma}_t \leftarrow \boldsymbol{\Sigma}_{t-1} + \boldsymbol{k}_t \boldsymbol{k}_t^\top$
  $\boldsymbol{o}_t \leftarrow \boldsymbol{S}_t \boldsymbol{q}_t$
end for
return $\{\boldsymbol{o}_t\}_{t=1}^T$

E.2 Chunk-Parallel Implementation

Chunk-Parallel Diagonal IA-DeltaNet. The implementation we use in practice mirrors the chunked DeltaNet forward pass of Flash Linear Attention,[1] replacing the write-side $\boldsymbol{k}_t$ with $\boldsymbol{u}_t$ and carrying the diagonal state $\boldsymbol{d}$ across chunks (Algorithm 3). Since the diagonal state $\boldsymbol{d}$ is updated by an element-wise prefix sum of squared keys, it is compatible with chunk-parallel training via associative scans.

[1] https://github.com/fla-org/flash-linear-attention/blob/main/fla/ops/delta_rule/naive.py

Algorithm 3 Chunk-parallel diagonal IA-DeltaNet (PyTorch-style pseudocode).

```
def chunk_diagonal_ia_deltanet(q, k, v, beta, S, d, chunk_size):
    # q, k: (B, H, T, d_k), v: (B, H, T, d_v)
    # S: recurrent memory state, using FLA row-vector convention
    # d: diagonal past-key overlap state, shape (B, H, d_k)

    v_beta = v * beta[..., None]
    k_beta = k * beta[..., None]

    q, k, v_beta, k_beta = chunk(q, k, v_beta, k_beta, chunk_size)
    outputs = []

    for q_i, k_i, v_i, k_beta_i in zip_chunks(q, k, v_beta, k_beta):

        # Diagonal IA write directions within the chunk.
        k_sq = k_i * k_i
        k_sq_prefix = cumsum(k_sq, dim=time)
        d_before = d[:, :, None, :] + shift_right(k_sq_prefix)

        p = k_i / clamp(d_before)
        s = sum(k_i * p, dim=-1, keepdim=True)
        u_star = p / clamp(s)

        # Chunked DeltaNet transform with write-side K^T -> U_star^T.
        T = -tril(k_beta_i @ transpose(u_star), diagonal=-1)
        T = solve_triangular(I - T, I, lower=True, unitriangular=True)

        v_bar = T @ v_i
        k_bar = T @ k_beta_i          # read side remains key-based
        delta = v_bar - k_bar @ S

        # Intra-chunk output and recurrent state update.
        A = tril(q_i @ transpose(u_star))
        o_i = q_i @ S + A @ delta
        S = S + transpose(u_star) @ delta

        # Carry diagonal overlap state to the next chunk.
        d = d + k_sq_prefix[:, :, -1]

        outputs.append(o_i)

    return concat(outputs, dim=time), S, d
```